```java
// joi/5/shapes/Line.java
//
//
// Copyright 2003 Bill Campbell and Ethan Bolker

/**
 * A Line has a length and a paintChar used to paint
 * itself on a Screen.
 *
 * Subclasses of this abstract class specify the direction
 * of the Line.
 *
 * @version 5
 */

public abstract class Line
{
    protected int  length;      // length in (character) pixels.
    protected char paintChar;   // character used for painting.

    /**
     * Construct a Line.
     *
     * @param length length in (character) pixels.
     * @param paintChar character used for painting this Line.
     */

    protected Line( int length, char paintChar )
    {
        this.length    = length;
        this.paintChar = paintChar;
    }

    /**
     * Get the length of this line.
     *
     * @return the length in (character) pixels.
     */

    public int getLength()
    {
        return length;
    }

    /**
     * Set the length of this line.
     *
     * @param length the new length in (character) pixels.
     */

    public void setLength( int length )
    {
        this.length = length;
    }

    /**
```

```java
     * Get the paintChar of this Line.
     *
     * @return the paintChar.
     */

    public char getPaintChar()
    {
        return paintChar;
    }

    /**
     * Set the paintChar of this Line.
     *
     * @param paintChar the new paintChar.
     */

    public void setPaintChar( char paintChar )
    {
        this.paintChar = paintChar;
    }

    /**
     * Paint this Line on Screen s at position (x,y).
     *
     * @param s the Screen on which this Line is to be painted.
     * @param x the x position for the line.
     * @param y the y position for the line.
     */

    public abstract void paintOn( Screen s, int x, int y );

    /**
     * Paint this Line on Screen s at position (0,0).
     *
     * @param s the Screen on which this Line is to be painted.
     */

    public void paintOn( Screen s )
    {
        paintOn( s, 0, 0 );
    }
}
```

```java
// joi/5/shapes/HLine.java
//
//
// Copyright 2003 Bill Campbell and Ethan Bolker


/**
 * An HLine is a horizontal Line.
 */

public class HLine extends Line
{
    /**
     * Construct an HLine having a paintChar and a length.
     *
     * @param length length in (character) pixels.
     * @param paintChar character used for painting this Line.
     */

    public HLine( int length, char paintChar )
    {
        super( length, paintChar );
    }

    /**
     * Paint this Line on Screen s at position (x,y).
     *
     * @param screen the Screen on which this Line is to be painted.
     * @param x      the x position for the line.
     * @param y      the y position for the line.
     */

    public void paintOn( Screen screen, int x, int y )
    {
        for ( int i = 0; i < length; i++ )
            screen.paintAt( paintChar, x+i, y );
    }

    /**
     * Unit test for class HLine.
     */

    public static void main( String[] args )
    {
        Terminal terminal = new Terminal();

        terminal.println( "Self documenting unit test of HLine.");
        terminal.println( "The two Screens that follow should match.");
        terminal.println();
        terminal.println( "Hard coded picture:");
        terminal.println( "++++++++++++++++++++++");
        terminal.println( "+xxxxxxxxxx          +");
        terminal.println( "+xxxxx               +");
        terminal.println( "+                    +");
        terminal.println( "+   *****            +");
        terminal.println( "+    1               +");
```

```java
        terminal.println( "+                    +");
        terminal.println( "++++++++++++++++++++++");
        terminal.println();

        terminal.println( "Picture drawn using HLine methods:");
        Screen screen = new Screen( 20, 6 );

        Line hline = new HLine( 10, 'x' );
        hline.paintOn( screen );

        hline.setLength(5);
        hline.paintOn( screen, 0, 1 );

        hline.setPaintChar('*');
        hline.paintOn( screen, 3, 3 );

        hline.setLength(1);
        hline.setPaintChar('1');
        hline.paintOn( screen, 4, 4 );

        screen.draw( terminal );
    }

}
```

```java
// joi/5/shapes/VLine.java
//
//
// Copyright 2003 Bill Campbell and Ethan Bolker

/**
 * A VLine is a vertical Line.
 */

public class VLine extends Line
{
    /**
     * Construct a VLine having a paintChar and a length.
     *
     * @param length length in (character) pixels.
     * @param paintChar character used for painting this Line.
     */

    public VLine( int length, char paintChar )
    {
        super( length, paintChar );
    }

    /**
     * Paint this Line on Screen s at position (x,y).
     *
     * @param screen the Screen on which this Line is to be painted.
     * @param x      the x position for the line.
     * @param y      the y position for the line.
     */

    public void paintOn( Screen screen, int x, int y )
    {
        for ( int i = 0; i < length; i++ )
            screen.paintAt( paintChar, x, y+i );
    }

    /**
     * Unit test for class VLine.
     */

    public static void main( String[] argv )
    {
        Terminal terminal = new Terminal();

        terminal.println( "Self documenting unit test of VLine.");
        terminal.println( "The two Screens that follow should match.");
        terminal.println();
        terminal.println( "Hard coded picture:");
        terminal.println( "+++++++++");
        terminal.println( "+xx     +");
        terminal.println( "+xx     +");
        terminal.println( "+xx     +");
        terminal.println( "+xx *   +");
        terminal.println( "+xx *1  +");
        terminal.println( "+x  *   +");
```

```java
        terminal.println( "+x  *   +");
        terminal.println( "+   *   +");
        terminal.println( "+       +");
        terminal.println( "+++++++++");
        terminal.println();

        terminal.println( "Picture drawn using VLine methods:");
        Screen screen = new Screen( 7,  9 );

        Line vLine = new VLine(  7, 'x' );
        vLine.paintOn( screen );

        vLine.setLength(5);
        vLine.paintOn( screen, 1, 0 );

        vLine.setPaintChar('*');
        vLine.paintOn( screen, 3, 3 );

        vLine.setLength(1);
        vLine.setPaintChar('1');
        vLine.paintOn( screen, 4, 4 );

        screen.draw( terminal );
    }

}
```

```java
// joi/5/shapes/ShapeOnScreen.java
//
//
// Copyright 2003 Bill Campbell and Ethan Bolker

// This file is used in one of the Chapter 5 exercises on shapes.

/**
 * A ShapeOnScreen models a Shape to be painted at
 * a given position on a Screen.
 *
 * @see Shape
 * @see Screen
 *
 * @version 5
 */

public class ShapeOnScreen
{
    private Shape shape;
    private int x;
    private int y;

    /**
     * Construct a ShapeOncreen.
     *
     * @param shape the Shape
     * @param x its x coordinate
     * @param y its y coordinate
     */

    public ShapeOnScreen( Shape shape, int x, int y )
    {
        this.shape = shape;
        this.x     = x;
        this.y     = y;
    }

    /**
     * What Shape does this ShapeOnScreen represent?
     *
     * @return the Shape.
     */

    public Shape getShape() {
        return shape;
    }

    /**
     * The current x coordinate of this ShapeOnScreen.
     *
     * @return the x coordinate.
     */

    public int getX() {
        return x;
```

```java
    }

    /**
     * The current y coordinate of this ShapeOnScreen.
     *
     * @return the y coordinate.
     */

    public int getY() {
        return y;
    }

    /**
     * Unit test.
     */

    public static void main( String[] args ) {
        ShapeOnScreen sos = new ShapeOnScreen( null, 5, 7);
        System.out.println("Shape: " + sos.getShape());
        System.out.println("x:     " + sos.getX());
        System.out.println("y:     " + sos.getY());
    }
}
```